Deep Learning Essentials

1. Machine Learning 101

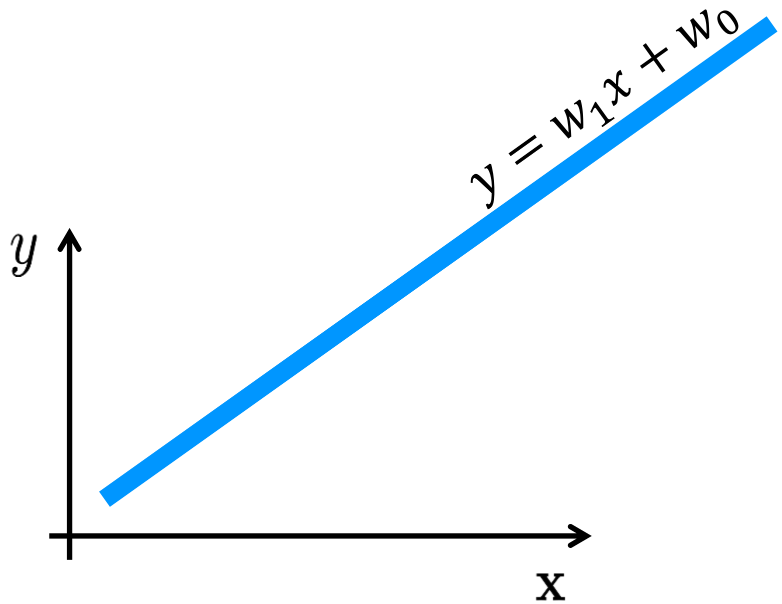
Models, learning, loss function

Lukáš Neumann

Adapted from B3B33UROB slides by Karel Zimmerman

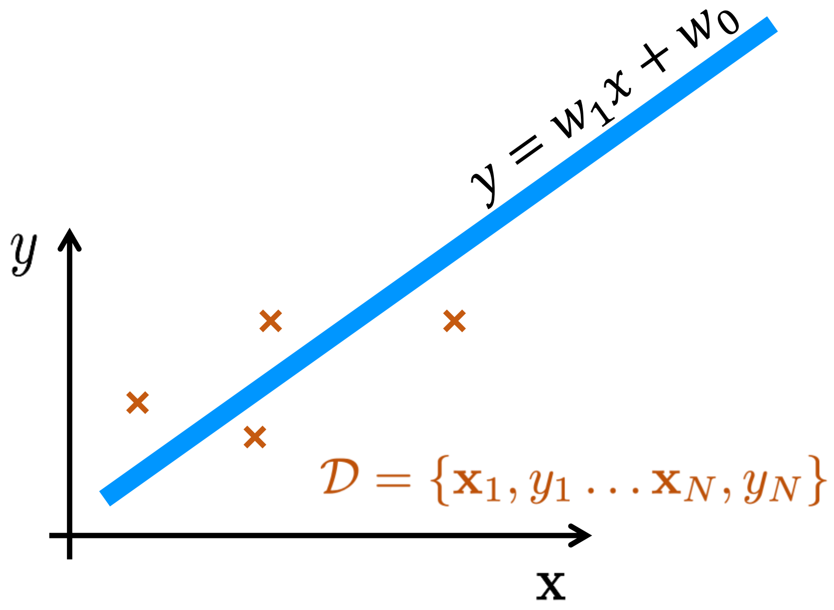
Motivating example: House price prediction

- TASK #1: Predict *market price* [€] of a house based on its *floor area* [m^2]
- We need to create a **model**
  - **Input:** floor area ... x
  - **Output:** price ... y
  - We look for some function f ... $\mathrm{y=f(x)}$
- How would you find such a function (=create the model)?
- The simplest model - linear function



Motivating example: House price prediction

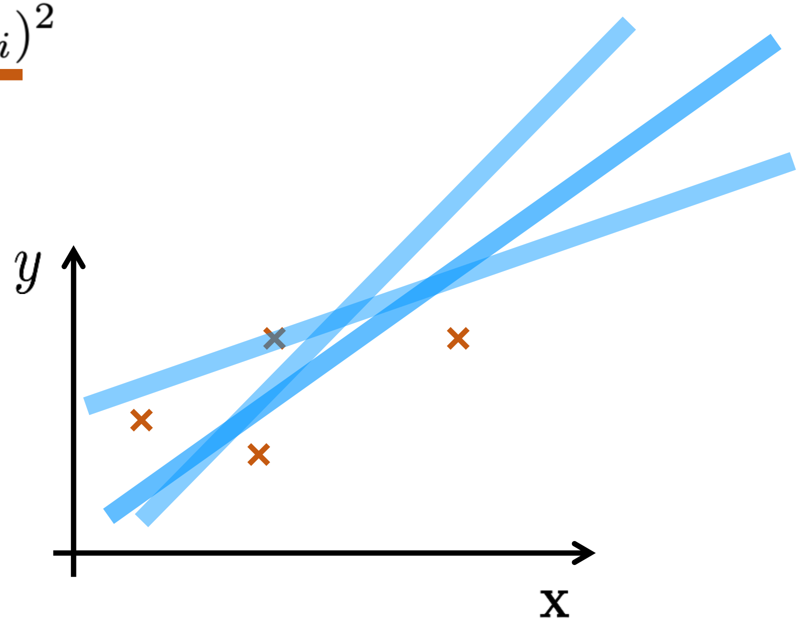
- **Model** = linear function
- In our case, assume the model has 2 parameters: w_0 and w_1
  - How to find their values?
  - Use known house floor area and the corresponding price as training data $\mathcal{D}$



Motivating example: House price prediction

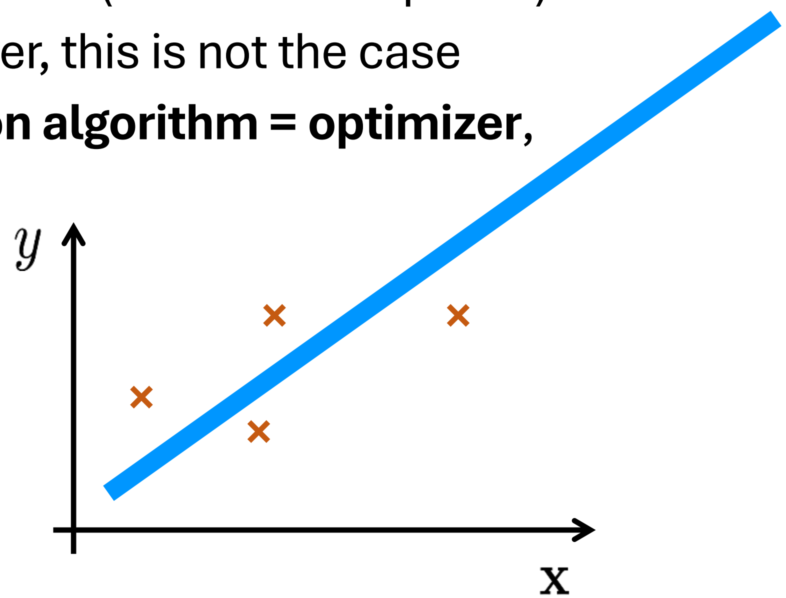
- **Model** = linear function
- **Training data** = known house floor area and their price
- How to tell which model parameters are better?
  - We need a criterion to compare different models
  - **Loss function**
    - Also known as: objective function, penalty function, cost function
  - In our case, let's use Mean Square Error (MSE)

$$\sum_i (\underline{w_1 x_i + w_0} - \underline{y_i})^2$$



Motivating example: House price prediction

- **Model** = linear function
- **Training data** = known house floor area and their price
- **Loss function** = Mean Square Error (MSE)
- We need an algorithm to find parameters which have the best (=lowest) loss function value $\arg\min_{\mathbf{w}} \sum_i (w_1 x_i + w_0 - y_i)^2$
  - For a linear function, we are lucky to know a closed-form solution to minimize Mean Square Error (linear least squares)
  - For most models however, this is not the case
  - We need an **optimization algorithm = optimizer**, e.g. Gradient Descent

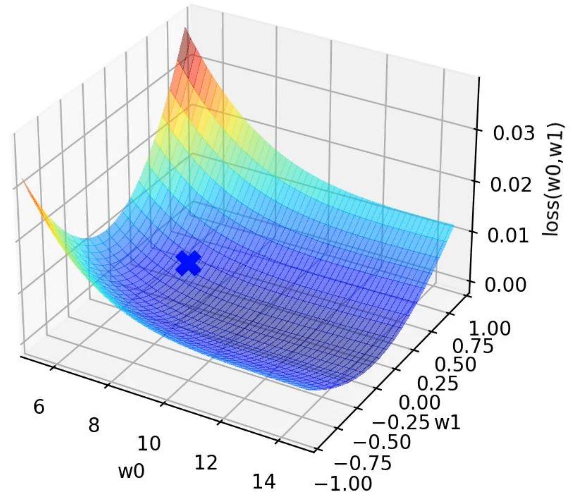


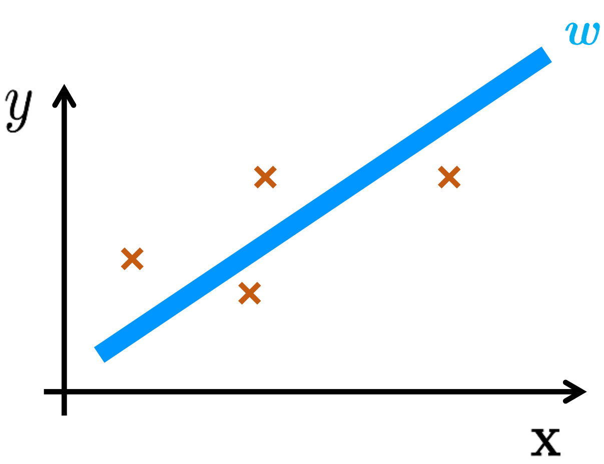
Motivating example: House price prediction

- **Model** = linear function
- **Training data** = known houses and their price
- **Loss function** = Mean Square Error (MSE)
- **Optimizer** = Gradient Descent (GD)
- Now let's put it together!

```
def train_model(x: np.ndarray, y: np.ndarray):
 w = np.array([-2.0, 2.0]) # init
 for _ in range(0, 10): # ten iterations
  loss = np.sum( (w[0] * x + w[1] - y)**2 )
  w = w - 0.1 * grad(loss, w) # update model

 return w
```





Model validation

- Now we have a model
- **But how well does the model work in practice?**
  - We need to validate the model on data not used during training = **test data**
  - Why we need independent data for testing?
    - We want to see how well the model generalizes on unseen data
    - Some learning algorithms have zero error on training data (nearest neighbor, decision trees, neural networks, ...)
- **How to get good testing data?**

GPT-5

Testing data should be:

- **Representative** – covers real-world scenarios and edge cases.
- **Balanced** – avoids bias toward common cases only.
- **Sufficiently sized** – enough samples to give reliable results.
- **Clean & consistent** – minimal errors, duplicates, or mislabeled samples.
- **Relevant** – matches your system's input requirements.

Model validation

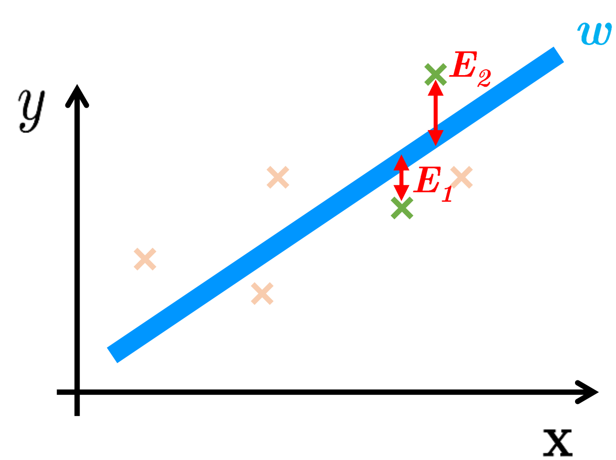
- Now we have a model
- **But how well does the model work in practice?**
  - We need to validate the model on <u>data not used during training</u> = **test data**
  - Why we need independent data for testing?
    - We want to see how well the model generalizes on unseen data
    - Some learning algorithms have <u>zero error</u> on training data (nearest neighbor, decision trees, neural networks, …)
- **How to get good testing data?**
  - Ideally, we want real-world data from production, this is not always possible
  - Split data into training and testing
  - Use synthetic/public data for training, real data for testing
  - "*Humans subconsciously select testing data that that are consistent with their proposed solution.*" (K. Popper)

Model validation

- Now we have a model and **test data**
- **We measure the prediction error on test data**
  - In our example we can use Mean Square Error (MSE, aka L2 norm)

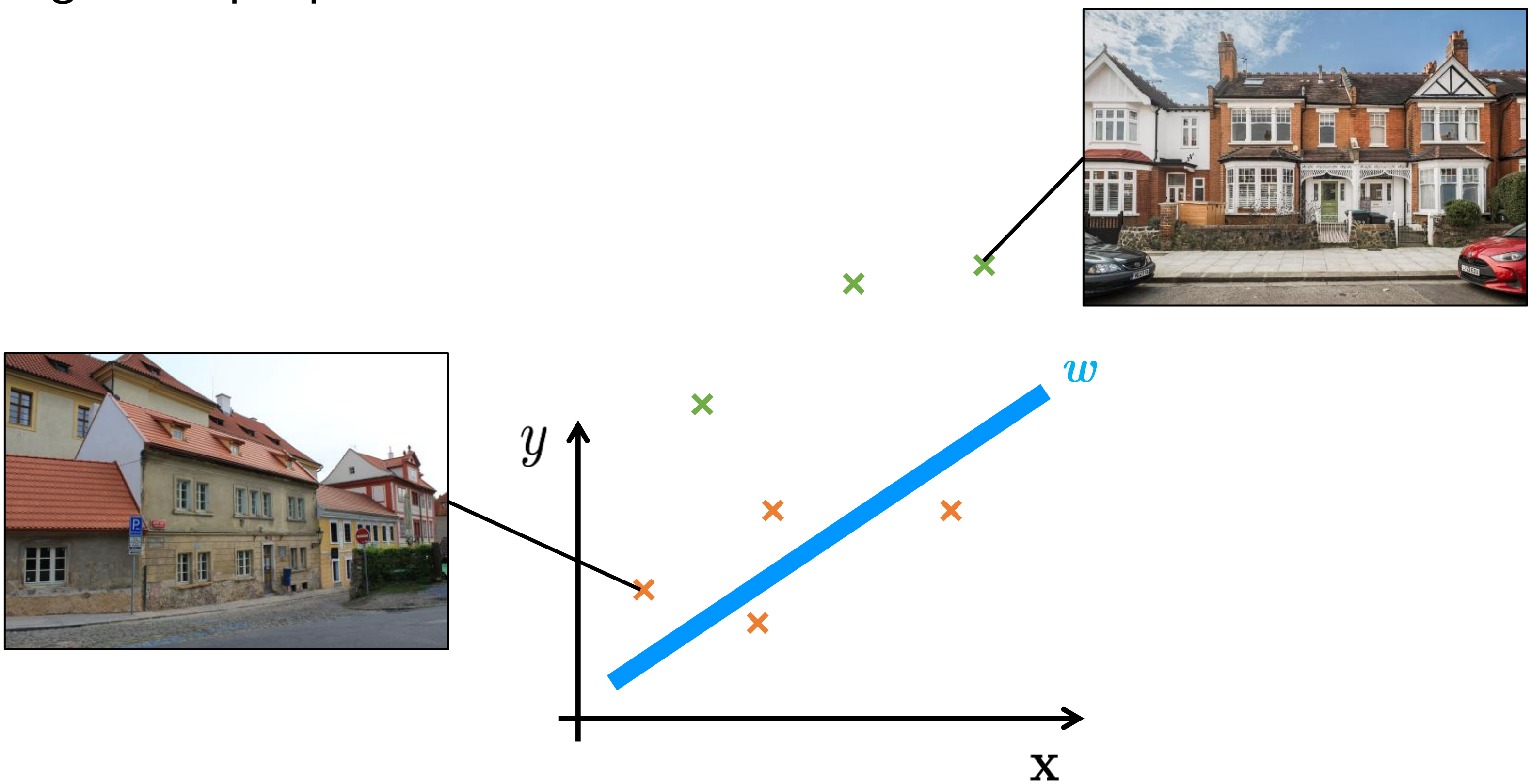
$$\sum_i (w_1 x_i + w_0 - y_i)^2$$

- What is the possible outcome of the validation on the test set
  - The error is low $\rightarrow$ great! **Or is it? Did we use good testing data?**
  - The error is high $\rightarrow$ **What could have gone wrong**?



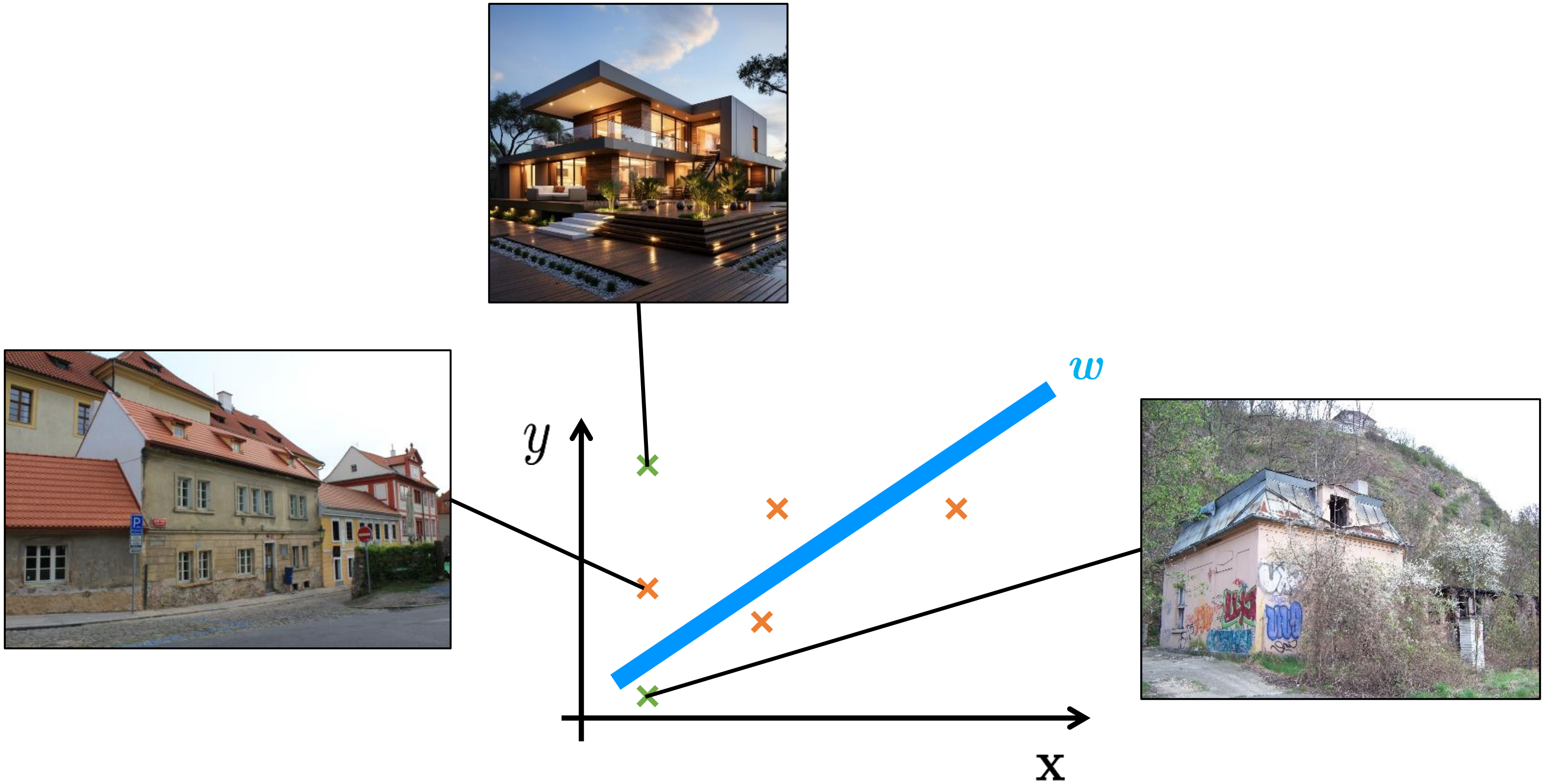
What can go wrong?

- **Training data have different distribution than test data**
- E.g. training data are house prices in Prague, but test data are from London
- Other examples: Day vs. night, summer vs. winter, young vs. old people



What can go wrong?

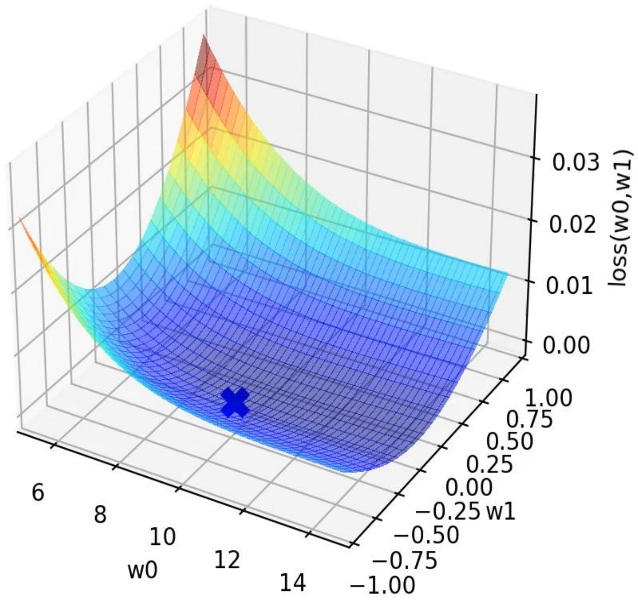
- **Inputs are not sufficient to predict the output**
- E.g. house price does not only depend on floor area, but house age, location etc.



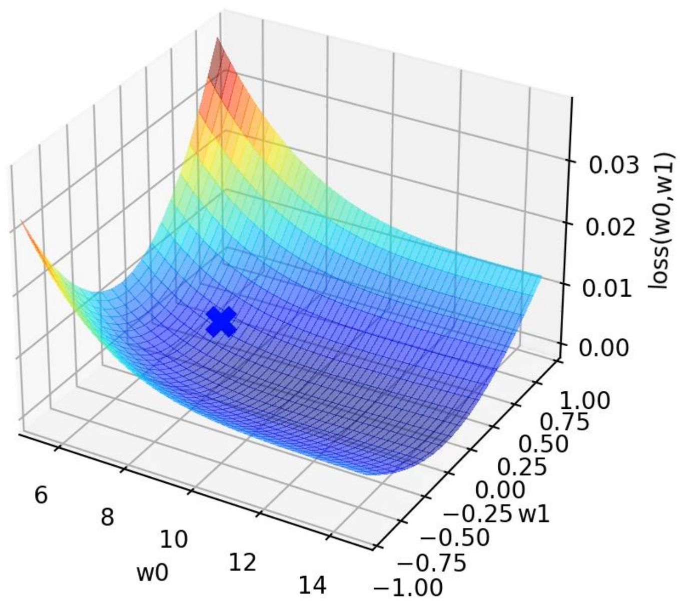
What can go wrong?

- **Learning fails to find good model parameters**
- Wrong hyper-parameters, stuck in a local optimum, bad initialization

High learning rate

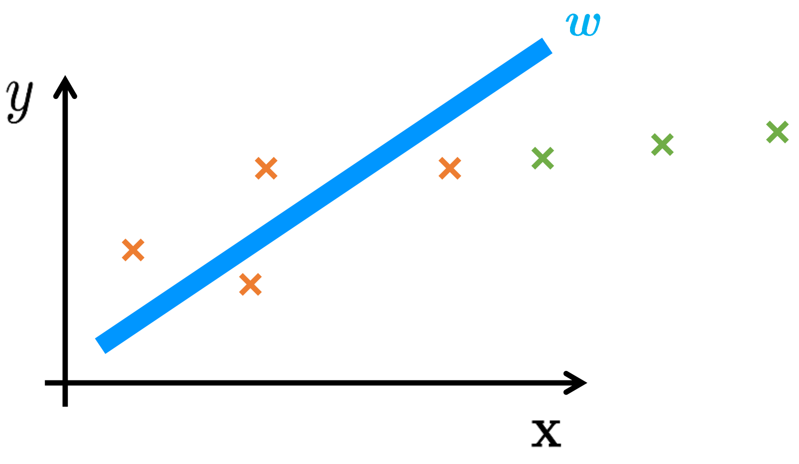


Reasonable learning rate



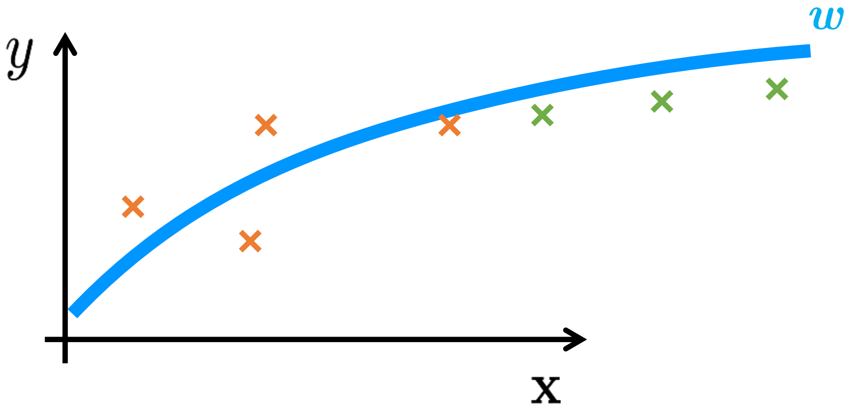
What can go wrong?

- **Inappropriate model**
- The model is too simple to fit the data = **underfitting**
- Bad generalization due to oversimplified model (linear function)



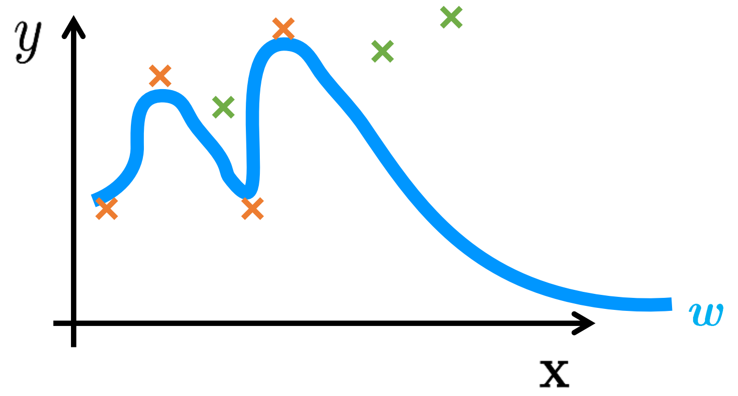
What can go wrong?

- **Inappropriate model**
- The model is too simple to fit the data = **underfitting**
- We can use more complicated model (logarithmic, polynomial, ...)



What can go wrong?

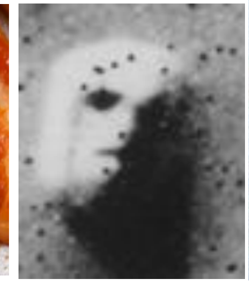
- **Inappropriate model**
- The model is too complex, e.g. 127^{th} degree polynomial
- It fits training data perfectly, but fails to generalize to new data = **overfitting**
- Do humans overfit?



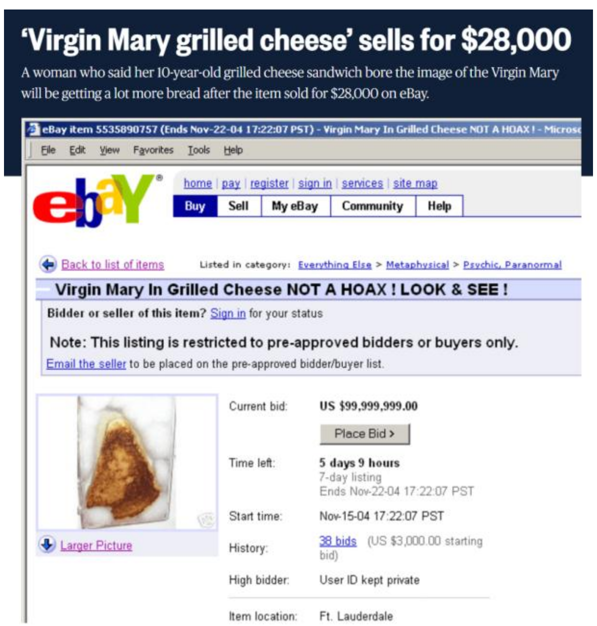
Do humans overfit?





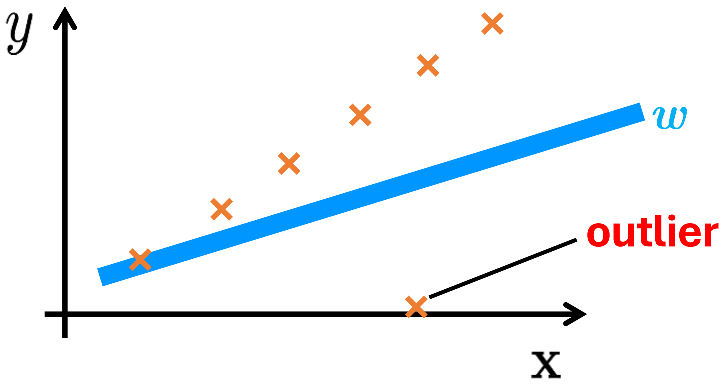




Apophenia (/æpoʊˈfiːniə/) is the tendency to perceive meaningful connections between unrelated things.



What can go wrong?

- **Inappropriate loss function**
- Mean Square Error (MSE) aka **L2 loss** function assumes noise in the training data is Gaussian (=random and symmetrical around true value)
- This assumption does not have to always be true, e.g. in presence of outliers
  - **Outlier** is a data point that differs significantly from other observations, due to measurement error or another anomaly
  - The L2 loss is extremely sensitive to outliers

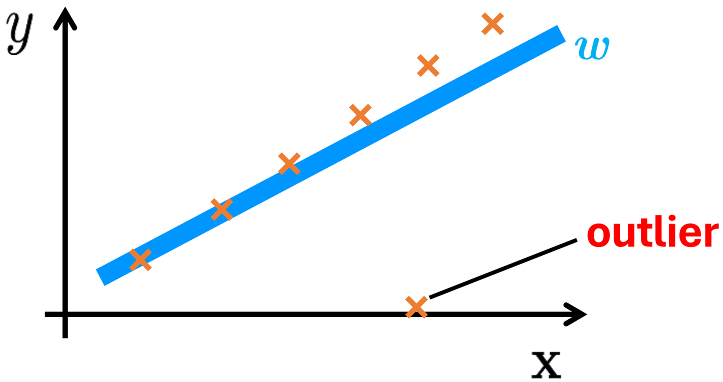


What can go wrong?

- **Inappropriate loss function**
- L1 loss is more robust to outliers than L2, but harder to optimize

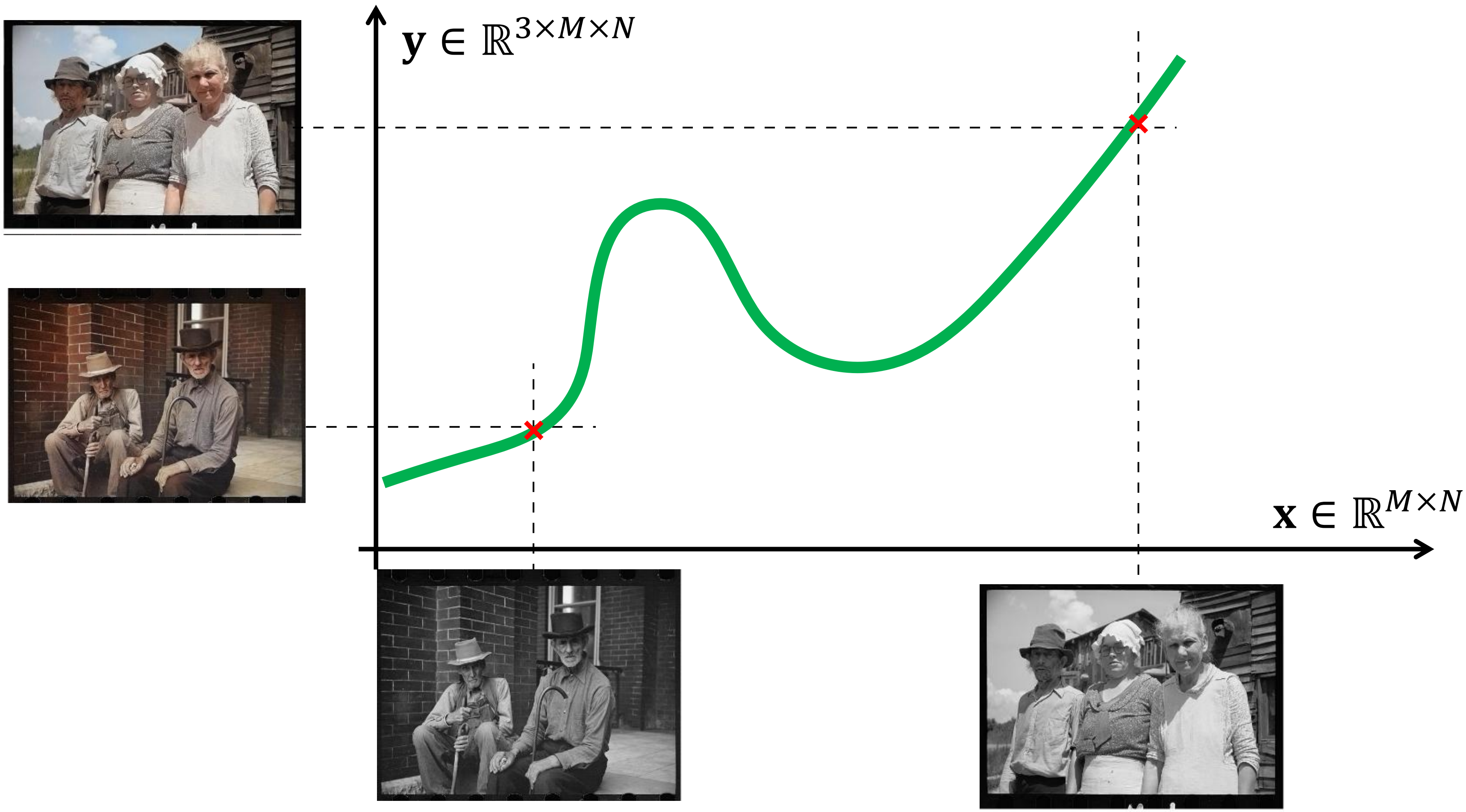
$$L1(y, y') = \sum_{i}^{N} |y_i - y_i'| \qquad\qquad L2(y,\ y') = \sum_{i}^{N} (y_i - y_i')^2$$

- There are other loss functions such as Huber, SmoothL1, ...



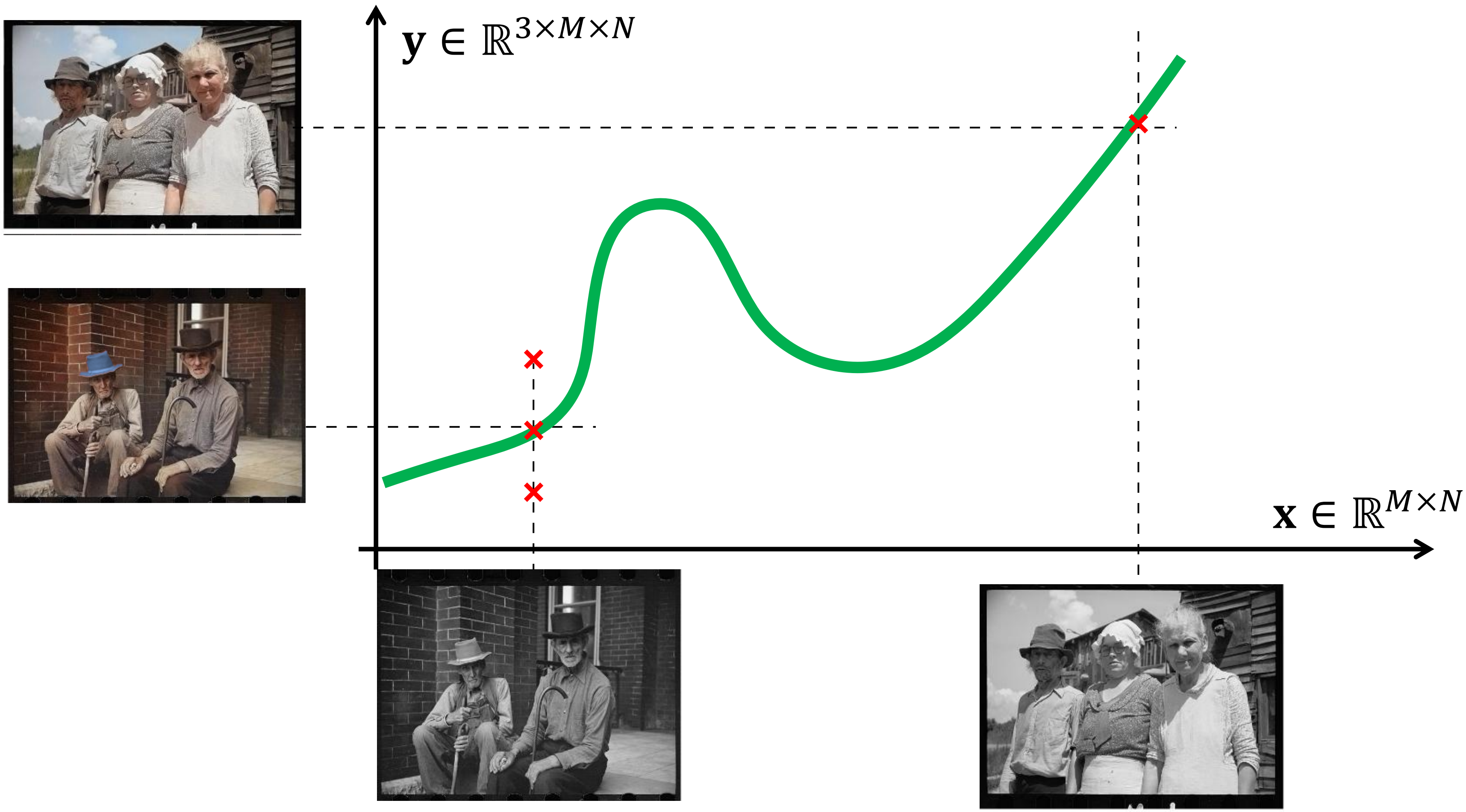
What can go wrong?

- **Inappropriate loss function**
- The loss function is completely inadequate for the problem
- For example, in **image colorization** calculating loss as per-pixel difference between color and gray-scale image



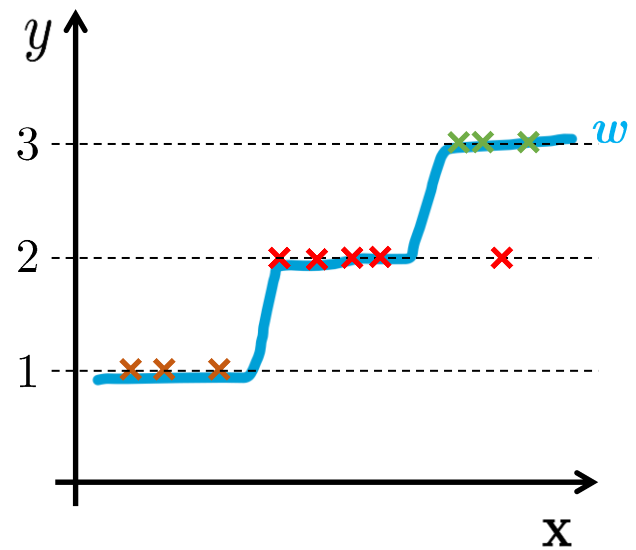
What can go wrong?

- **Inappropriate loss function**
- The loss function is completely inadequate for the problem
- For example, in **image colorization** calculating loss as per-pixel difference between color and gray-scale image



Is regression model always the best choice?

- TASK #2: Predict what type of building material was used based on thermal conductivity (heat transfer) measurements.
- We could assign each material a number (**wood** = 1, **brick** = 2, **concrete** = 3, …)
- What is problematic about this approach?
  - **Enforced ordering** (mistaking wood for brick has lower penalty than mistaking wood for concrete)
  - Model cannot output “either wood or concrete, but definitely not brick”
  - Sensitive to misclassified training samples

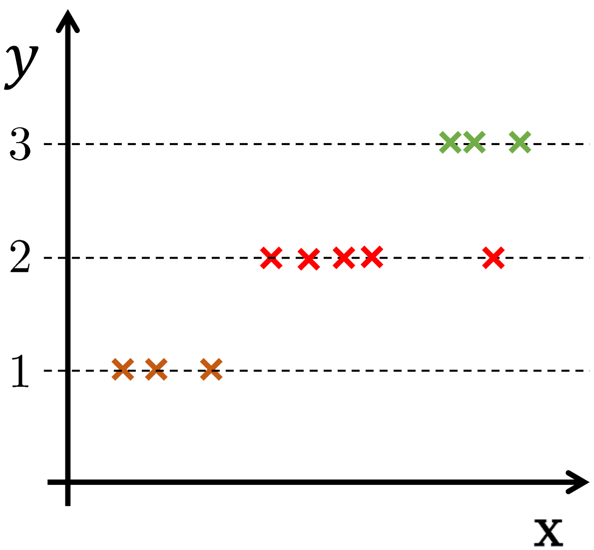


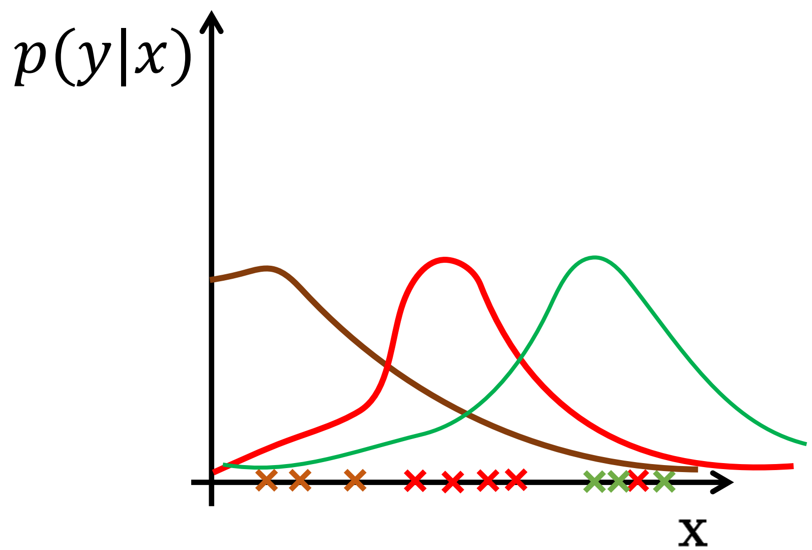
Classification models

- TASK #2: Predict *what type of building material was used* based on *thermal conductivity* (heat transfer) measurements.
- Instead, we model each class independently (e.g. 3 classes → 3 functions)

  $p(y = 1|x), p(y = 2|x), p(y = 3|x)$

  - Each function predicts **conditional** class probability
  - For given measurement they sum to one $\sum_{\mathrm{k}} p(y = \mathrm{k}|x) = 1$



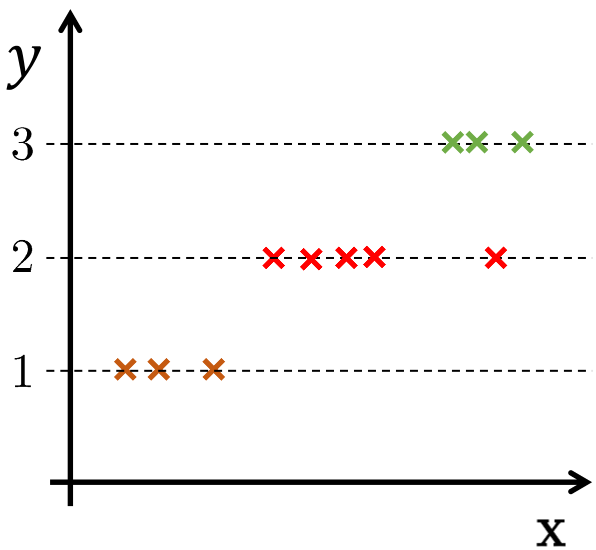


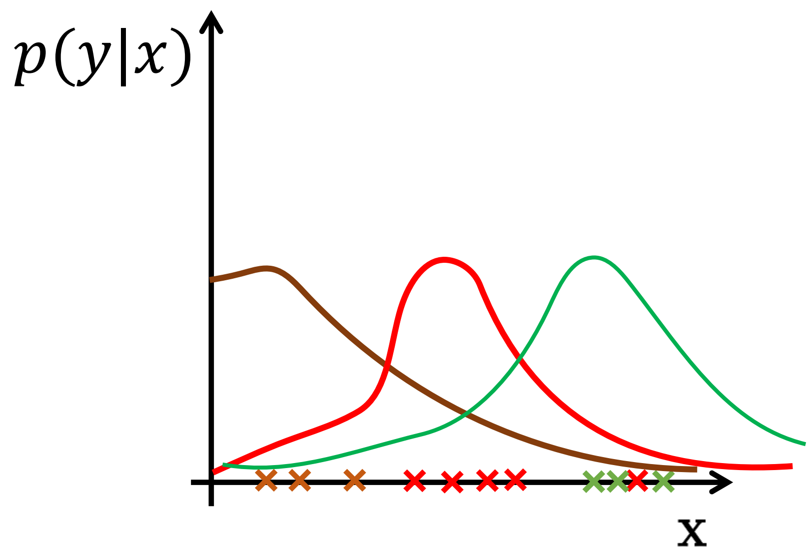
Classification models

- TASK #2: Predict *what type of building material was used* based on *thermal conductivity* (heat transfer) measurements.
- Instead, we model each class independently (e.g. 3 classes → 3 functions)

  $p(y = 1|x), p(y = 2|x), p(y = 3|x)$

  - Each function predicts class probability
  - For given measurement they sum to one $\sum_{\mathrm{k}} p(y = \mathrm{k}|x) = 1$
- **How to find such probabilities?**
  - Come to next lecture!





Competencies gained for the test

- What is a model?
- Learning (loss, training data, optimization procedure)
- Model evaluation
- Test data selection
- What can go wrong?
  - Test data have different distribution than training data
  - Inputs do not allow to predict outputs
  - Learning fails to find good model parameters
  - Inappropriate model, under- and over-fitting
  - Inappropriate choice of loss function
- Regression vs classification